Las clases que he identificado son:

Usuario, CuentaBancaria, Factura, Serie, Artista, Categoria, Capitulo, Temporada y Visualizacion. Hay una clase intermedia que la he introducido para cumplir con el domain driven design (la clase CapituloID)

Los tipos de estas clases son los siguientes:

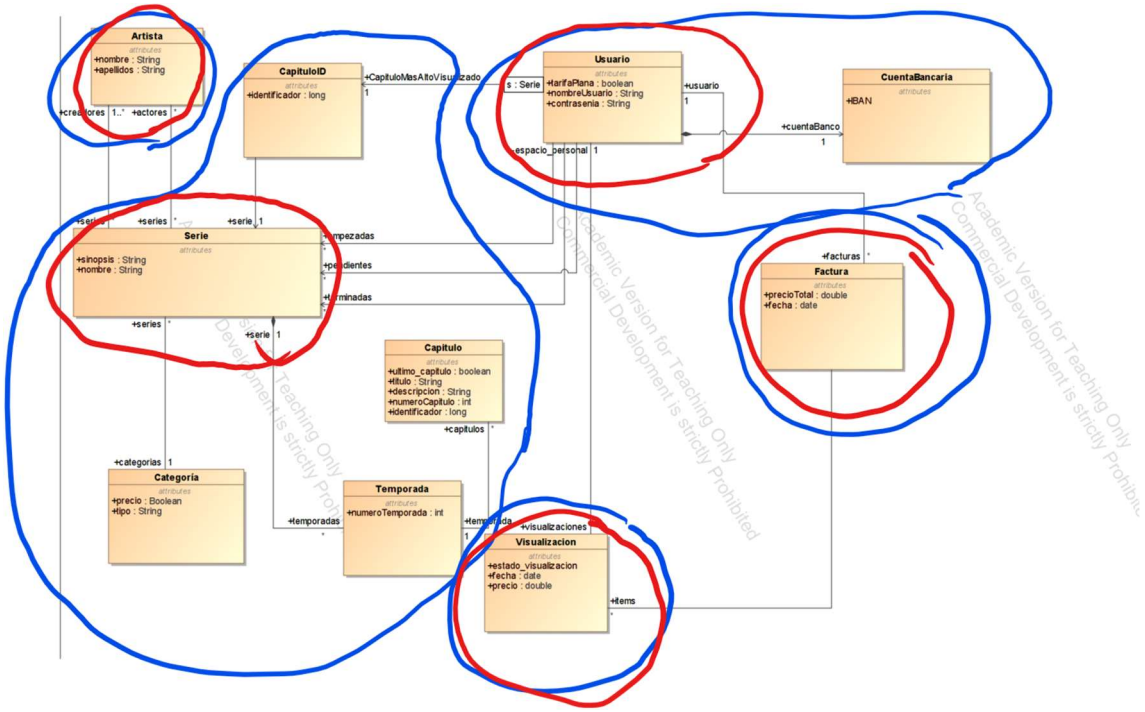
- **Entity:** Usuario, Factura, Serie, Temporada, Capitulo y Visualización.
  - Usuario: Es el elemento central de la plataforma, por lo cual queremos poder monitorizarlo y ser capaces de distinguir entre diferentes usuarios ya que cada uno va a tener acceso a diferentes “features” que ofrezca el sistema dependiendo del tipo de usuario que sea.
  - Factura: Cada factura es única por lo cual es necesario poder diferenciarlas unas de otras (no puede tener varias facturas “iguales” entre varios usuarios).
  - Serie: Cada serie también es única y es un elemento fundamental dentro de una plataforma de streaming, es decir, nos importa la entidad.
  - Temporada: Así mismo, cada temporada de una determinada serie también será única.
  - Capítulo: Cada capítulo dentro de una Serie también va a ser único (así como también dentro de su temporada), a pesar de que se llegue a dar el caso de que un capítulo tenga atributos similares con otro capítulo (como el número de capítulo), va a ser necesario poder identificar únicamente a cada capítulo para referenciarlo ínvidamente frente a otros.
  - Visualización: Cada visualización va a representar a una instancia única de cierto contenido visto por un Usuario, y para hacer seguimiento de estos, pues debe tener una entidad propia para poder distinguirlo (en vez de ser solo un puñado de datos sin referencia única)
- **Value object:** CuentaBancaria, Artista, Categoria y CapituloID.
  - CuentaBancaria: perfectamente puede haber metido este valor como un String dentro de la clase Usuario, pero por flexibilidad la he puesto como una clase ya que solo nos importa su contenido y no su identidad propia, ósea, solo interesa como un dato asociado al usuario.
  - Categoría: Pues también perfectamente pude haberla puesto como un atributo de la clase Serie pero por flexibilidad se ha sacado y puesto como una clase, y los valores de las distintas categorías se podrían traducir a un enumerado, del cual solo nos importa su valor y es por eso que es un value object, es decir, las categoría simplemente son etiquetas para clasificar el tipo de Serie.
  - Artista: Una vez más también se pudo haber puesto como un String dentro de la clase Serie, pero por flexibilidad se ha puesto como una clase de la cual solo nos importa su valor, aunque un artista represente a una persona real, según como está descrito el sistema, no es

necesario hacer un seguimiento de una identidad única, solo interesan ciertos valores de éste (como el atributo nombre).

- CapituloID: Es la clase intermedia la cual fue introducida para cumplir el domain driven design, esta clase simplemente nos interesa su valor más que sea única y monitorizable, a través de esta clase vamos a mejorar la eficiencia de acceso a saber el último capítulo visto por el usuario de una serie y es por eso por lo que solo nos interesa su valor.

Los aggregates y aggregates root:

En la siguiente imagen en azul he marcado los aggregates y dentro de estos he marcado en rojo el aggregate root:



Justificación

- Serie: Es uno de los elementos centrales de una plataforma de streaming, pero una Serie por sí sola no tiene sentido sin sus temporadas y capítulos por lo que estás van a formar parte de su aggregate, además una serie va a tener una categoría. Entonces, una Temporada pertenece a una única serie y no tiene sentido su existencia por si sola, luego un Capitulo no se puede acceder sin su temporada y por lo tanto sin su serie y por último, una Categoria no tiene entidad propia, simplemente es un value object asociado a la Serie.
  La clase CapituloID la he agregado para poder cumplir el domain driven design (ya que antes Usuario tenía un mapa de capítulos para poder hacer una búsqueda más eficiente sobre el último capítulo visto, pero como eso me generaba una referencia de un root, ósea usuario, a un no root de otro

aggregate, ósea Capitulo, tuve que meter la clase intermedia CapituloID), dicha clase al ser algo artificial sobre la clase Capitulo, la he metido en el aggregate de Serie.

- Usuario: Es la entidad principal para representar a personas en el sistema, y cada Usuario va a tener asociado una CuentaBancaria para poder realizar sus respectivos pagos, pero dicha cuenta solo tiene sentido en el contexto del usuario, no accederemos de forma independiente a esta.
- Factura: Representa un cobro único que va a tener su propia identidad en el sistema y no depende de otras entidades en el sistema (por ejemplo, si dependiera de usuario, al borrar el usuario también se borrarían las facturas, algo que no sería deseable porque como empresa es necesario poder justificar los ingresos si hacienda lo pidiera, por ejemplo).
- Visualización: Representa la acción de un usuario de ver un contenido específico, y es una entidad única que puede ser referencia independientemente para llevar a cabo ciertas monitorizaciones (por ejemplo, para darle recomendaciones a los usuarios o para métricas generales sobre el rendimiento de la plataforma).
- Artista: Es una persona asociada a una serie (ya sea como actor o director). Tiene identidad única en el sistema y puede estar vinculada a múltiples series por lo que no depende de ninguna otra entidad para existir.